Eğitim Teknolojisi

kuram ve uygulama

Kış 2017
Cilt 7
Sayı 1

Winter 2017
Volume 7
Issue 1

Educational Technology

theory and practice

ISSN: 2147-1908

EĞİTİM TEKNOLOJİSİ KURAM VE UYGULAMA / EDUCATIONAL TECHNOLOGY THEORY AND PRACTICE

Cilt 7, Sayı 1, Kış 2017
Volume 7, Issue 1, Winter 2017

Genel Yayın Editörü / Editor-in-Chief: **Dr. Halil İbrahim YALIN**
Yardımcı Editör / Co-Editor: **Dr. Tolga GÜYER**

Sorumlu Yazı İşleri Müdürü / Publisher Editor: **Dr. Sami ŞAHİN**
Redaksiyon / Redaction: **Dr. Tolga GÜYER**
Dizgi / Typographic: **Dr. Tolga GÜYER**
Sayfa Tasarımı / Page Design: **Dr. Tolga GÜYER**
Kapak Tasarımı / Cover Design: **Dr. Bilal ATASOY**
İletişim / Contact Person: **Dr. Aslıhan KOCAMAN KAROĞLU**

Dizinlenmektedir / Indexed in: **ULAKBİM Sosyal ve Beşeri Bilimler Veritabanı, Türk Eğitim İndeksi, ASOS Sosyal Bilimler İndeksi**

Editör Kurulu / Editorial Board*

* Liste isme göre alfabetik olarak oluşturulmuştur. / List is created in alphabetical order

Hakem Kurulu / Reviewers*

* Liste isme göre alfabetik olarak oluşturulmuştur. / List is created in alphabetical order.

İletişim Bilgileri / Contact Information

İnternet Adresi / Web: http://dergipark.ulakbim.gov.tr/etku/
E-Posta / E-Mail: tguyer@gmail.com
Telefon / Phone: +90 (312) 202 17 38
Belgegeçer / Fax: +90 (312) 202 83 87
Adres / Adress: Gazi Üniversitesi, Gazi Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü,
06500 Teknikokullar - Ankara / Türkiye

EĞİTİM TEKNOLOJİSİ Kuram ve Uygulama
Cilt:7 Sayı:1 Yıl:2017

Makale Geçmişi / Article History
Alındı/Received: 14.07.2016
Düzeltme Alındı/Received in revised form: 22.01.2017
Kabul edildi/Accepted: 23.01.2017

COMPUTER PROGRAMMING SELF-EFFICACY SCALE (CPSES) FOR SECONDARY SCHOOL STUDENTS: DEVELOPMENT, VALIDATION AND RELIABILITY

Volkan KUKUL[1] , Şahin GÖKÇEARSLAN[2]*, Mustafa Serkan GÜNBATAR[3]

Abstract

Computer programming has been included in the curriculum of K12 education around the world and this has necessitated a tool for the assessment of the computer programming self-efficacy. Thus, this study aims to suggest the necessary scale for the field. In the scale development, the steps of classical measurement theory were applied. Following the expert review, the item pool was conducted with 233 students in a public secondary school which provides education to the age group of 12-14 in the school year 2014-2015. As a result of the study, a unidimensional 5-point Likert scale of 31 items was obtained. The factor loads varied between 0.47 and 0.71 and the explained variance rate was 41.15%. In the analysis of the internal consistency, sufficient values were found; the Cronbach alpha as 0.95 and the equivalent halves method result as 0.96. For the construct validity, exploratory and confirmatory factor analysis were applied and the result showed that the scale is valid and reliable.

Keywords: Computer Programming, Teaching Computer Programming, Self-Efficacy

ORTAOKUL ÖĞRENCİLERİ İÇİN PROGRAMLAMA ÖZYETERLİK ÖLÇEĞİ: GELİŞTİRME, GEÇERLİK VE GÜVENİRLİK

Öz

Bilgisayar programlama, son yıllarda tüm dünyada K-12 eğitim müfredatlarında yer almaya başlamıştır ve programlama öz-yeterliğinin ölçülmesi için bir araç geliştirilmesine ihtiyaç duyulmuştur. Bu çalışmanın amacı bu ihtiyacı gidermek adına alana katkı sağlamaktır. Aracın geliştirilmesinde klasik ölçme teorisinin basamakları kullanılmıştır. Çalışma 2014-2015 eğitim öğretim yılının bahar döneminde bir devlet okulunda yaşları 12-14 arasında değişen 233 öğrenci ile yürütülmüştür. Çalışmanın sonucunda 31 maddeden oluşan tek faktörlü ölçme aracı ortaya çıkmıştır. Ölçme aracındaki maddelerin madde yükleri 0.47 ile 0.71 arasında

[1] Ar. Gör., Gazi Üniversitesi, volkankukul@gazi.edu.tr
[2] Dr., Gazi Üniversitesi, sgokcearslan@gazi.edu.tr *Corresponding Author
[3] Yrd.Doç.Dr., Van 100.yıl Üniversitesi, msgunbatar@gmail.com

değişmektedir ve ölçeğin açıkladığı toplam varyans %41.15'tir. Ölçeğin iç tutarlılığını belirlemek amacıyla yapılan analizlere göre Cronbach alfa katsayısı 0.95, iki yarı metodu sonuçları ise 0.96 çıkmıştır ve bu sonuçlar ölçme aracının iç tutarlılığının yüksek olduğunu göstermektedir. Ölçeğin yapı geçerliğini belirlemek amacıyla açımlayıcı ve doğrulayıcı faktör analizleri uygulanmış ve analiz sonuçlarına göre ölçeğin geçerli ve güvenilir olduğu sonucuna varılmıştır.

Anahtar Kelimeler: Bilgisayar Programlama, Bilgisayar Programlama Eğitimi, Öz-Yeterlilik

Geniş Özet

Günümüzde, bilişim teknolojilerindeki gelişmelerle insanların problemlere çözüm arayışları farklılaşmıştır. Uzun sürede bitirilebilecek bir iş ya da görev teknolojiyi kullanarak kısa sürede bitirebilmektedir. Dijital teknolojinin içerisinde büyüyen çocukların, sadece o teknolojiyi kullanmaları değil, gerekirse yeni teknolojiler üreterek üst düzey düşünme becerilerinin gelişmesi beklenmektedir (Kalelioğlu, 2015). Bu üst düzey becerilerden bir tanesi de Bilgi İşlemsel Düşünmedir (Computational Thinking) (Philips, 2009; Wing 2010). Bilgi İşlemsel Düşünme, sadece bilgisayar bilimcilerinin değil tüm insanların sahip olmaları gereken bir beceri olarak görülmektedir (Korkmaz, Çakir, & Özden, 2017; Wing, 2006; Wing, 2008; Wing, 2010).

Bilgi İşlemsel Düşünme becerisinin öğrencilere kazandırılmasında sık kullanılan yöntemlerden bir tanesi, "Başlangıç Öğrenme Ortamları" olarak değerlendirilen görsel programlama araçlarıyla bilgisayar programlama öğretimidir (Weinberg, 2013). Programlama becerisi yaratıcı düşünme, problem çözme, mantıksal çıkarım gibi üst düzey düşünme becerilerinin geliştirilmesine olanak tanımaktadır (Fesakis & Serafeim, 2009; Fessakis, Gouli, & Mavroudi, 2013; Kay & Knaack, 2005). Bilgisayar programlama becerisinin öğrencilere sağladığı katkı, eğitimcilerin ve araştırmacıların ilgisini çekmiş (Gökçearslan & Alper, 2015), bu doğrultuda Avrupa ve Amerika'da erken yaşlar için bilgisayar programlama öğretimine yönelik ders ve etkinlikler üzerine yapılan çalışmalar artmıştır (Grover & Pea, 2013; Kafai & Burke, 2013). Birçok ülke erken yaşlar için bilgisayar programlamayı ulusal programlarına entegre etmeye başlamışlardır (Kalelioğlu, 2015).

Geçmişte öğrencilere bilgisayar programlamanın öğretilmeye çalışılıp, sadece az sayıdaki öğrencinin başarılı olması (Resnick et al., 2009), "bu sefer de aynı sorunla karşı karşıya kalınır mı?" sorusunu akla getirmektedir. Bunun için öğrencilerin bilgisayar programlamaya yönelik düşünceleri ve bilgisayar programlamadaki başarılarının değerlendirilmesi gerekmektedir. Öğrencilerin öz-yeterlilik düzeylerinin belirlenmesi, başarıları hakkında yorum yapabilmek için önemli bir faktör olarak görülmektedir (Aşkar & Davenport, 2009; Anastasiadou & Karakos, 2011). Farklı konu alanlarında öz-yeterlilik düzeyini ölçmek için pek çok araştırma olmasına rağmen, bilgisayar programlama öz-yeterliliğini ölçmek için yapılmış çalışma sayısının sınırlı olduğu ifade edilmektedir (Aşkar & Davenport, 2009). Yapılan çalışmaların genellikle lise ve üniversite düzeyinde oldukları görülmektedir (Aşkar & Davenport, 2009; Korkmaz & Altun, 2014; Mazman & Altun, 2013; Ramalingam & Wiedenbeck,1998). Bu bağlamda bu çalışmanın odak noktasını erken yaşlardaki öğrencilerin öz-yeterlilik düzeylerini belirlemek için "Ortaokul öğrencileri için Programlama Öz-yeterlilik Ölçeğinin" geliştirilmesi oluşturmaktadır.

Bu araştırmada ortaokul öğrencilerinin programlama öz-yeterlik düzeylerini ölçmek amacıyla bir ölçek geliştirmek istendiği için, 2014-2015 öğretim yılında Ankara'daki bir devlet ortaokulunda öğrenim gören toplam 233 öğrenci çalışma grubunu oluşturmuştur. Grubun % 53.6'sını (n=125) kız öğrenciler, % 46.4'ünü (n=108)erkek öğrenciler oluşturmuştur. Ayrıca öğrencilerin % 19.3'ü (n=45) 5. sınıf, % 59.2'si (n=138) 6. Sınıf ve % 21.5'i (n=50) 7. sınıfa devam etmektedir.

Kaiser-Meyer-Olkin (KMO) katsayısı faktör analizi yapmak için örneklem sayısının yeterli olup olmadığını belirlemede kullanılan bir istatistiksel yöntemdir (Kan & Akbaş, 2005). Bu amaçla KMO değeri belirlenmiştir ve 0.949 olarak karşımıza çıkmıştır. Faktör analizi yapılabilmek için en düşük KMO değerinin 0.60 olması önerilmektedir (Özel, Timur, Timur ve Bilen, 2013; Pallant, 2001). İkinci olarak Bartlett Sphericity testine bakılmıştır ($\chi 2$ = 3532.449, p.=0.000). Verilerin çok değişkenli normal dağılım gösterdiği Barlett Sphericity testi sonucu elde edilen Kay-kare test istatistiğinin anlamlı çıkması ile anlaşılmaktadır (Kan & Akbaş, 2005). Bu sonuçlar, toplanan verilerle açımlayıcı faktör analizi yapılabileceğine işaret etmiştir.

Yapılan açımlayıcı faktör analizi sonucunda, başlangıçta 33 maddelik ölçekten birden fazla faktöre benzer yük veren iki madde çıkartılmıştır. Ölçeğin nihai formunda 31 madde bulunmaktadır. Tek faktör altında değerlendirilen ölçek % 41.15 varyans açıklama yüzdesine sahiptir. Sosyal bilimler araştırmaları için bu oran kabul edilebilir bir düzeye karşılık gelmektedir (Büyüköztürk, 2010; Çokluk, Şekercioğlu & Büyüköztürk, 2010; Hair, Anderson, Tatham & Black, 1998).

Açımlayıcı Faktör Analizi (AFA) sonucu elde edilen faktör yapısına ilişkin modelin uygunluğu Doğrulayıcı Faktör Analizi (DFA) ile test edilmiştir. Modelin uygunluğuna ilişkin analiz sonuçlarına göre; X2 / df=1.84; RMSEA değeri 0.06; NFI değeri 0.95; NNFI değeri 0.98; RMR değeri 0.068; CFI değeri 0.98; IFI değeri 0.98; GFI değeri 0.82 ve AGFI değeri 0.79 dur. Bu analiz sonuçlarına göre model uyum değerlerinden bazılarının kabul edilebilir düzeyde olmadıkları görülmüştür. Modifikasyon önerileri gerçekleştirilmiştir. Bu modifikasyonlar sonucunda X2/df=1.47; RMSEA değeri 0.045; NFI değeri 0.96; NNFI değeri 0.99; RMR değeri 0.061; CFI değeri 0.99; IFI değeri 0.99; GFI değeri 0.85 ve AGFI değeri 0.83 olarak tespit edilmiştir.

Cronbach alpha güvenirlik analizi sonucunda ölçeğin güvenirlik katsayısı 0.95 olarak oldukça yüksek değerde çıkmıştır (Özdamar, 1999). Ölçek maddeleri tek ve çift maddeler olmak üzere iki yarıya bölünmüş ve eşdeğer yarılar (testi yarılama) yöntemiyle de güvenirlik analizi gerçekleştirilmiştir. Testin tümüne ait güvenirlik katsayısı Spearman-Brown yöntemi kullanılarak bulunabilir (Ellez, 2009). Bu noktadan hareketle testin tamamına ilişkin Spearman Brown yöntemi ile elde edilen güvenirlik katsayısı r = 0.966 bulunmuştur. Testin birinci ve ikinci yarısı arasındaki ilişki istatistiksel açıdan $p<0.01$ düzeyinde pozitif yönde anlamlı bulunmuştur.

Tek faktörlü yapı gösteren bilgisayar programlama öz-yeterlik ölçeğini öğretmen ve araştırmacılar özellikle son zamanlarda yaygın biçimde çocuklara programlama becerisi kazandırmak için kullanılan Scratch, Logo, Alice vb. programların öğretimi sürecinde öğrencilerin programlama öz-yeterlik düzeyini ölçmek için kullanabilirler.

Introduction

Today's developments in information technologies have caused human beings to seek different solutions to their fundamental problems. Technology can enable individuals to finish a task that would previously have taken a long time in as short a time as possible. Children growing up within the digital technology age are expected to not only use that technology, but also to produce new technologies and develop high-level thinking skills, if necessary (Kalelioğlu, 2015). One of these high-level skills is Computational Thinking (Philips, 2009; Wing 2010). Computational Thinking is considered as a skill that should be possessed not only by computer scientists, but everyone (Korkmaz, Çakir, & Özden, 2017; Wing, 2006; Wing, 2008; Wing, 2010).

One of the methods frequently used in teaching the skill of computational thinking to students is computer programming teaching via visual programming instruments that can be seen and evaluated as 'Initiative Learning Environments' (Weinberg, 2013). Computer programming skills contribute to the development of other high-level skills like problem-solving, logical inference and creative thinking (Fesakis & Serafeim, 2009; Fessakis, Gouli, & Mavroudi, 2013; Kay & Knaack, 2005). Trainers and researchers have become aware of the contributions made to students by having the skill of computer programming (Gökçearslan & Alper, 2015), which has resulted in the increase of courses and activities aimed at computer programming teaching for young ages in both Europe and the United States (Grover & Pea, 2013; Kafai & Burke, 2013). A number of countries have started to integrate computer programming for young people into their national programs (Kalelioğlu, 2015).

Alongside the development of computers, one of the aims was to teach all children computer programming methods (Resnick et al., 2009). However, the difficulties experienced by students while writing programs on a program-compiler and the use of uninteresting activities in computer programming teaching (Resnick et al., 2009) caused students to consider computer programming a difficult task (Aşkar & Davenport, 2009; Caspersen & Kolling 2009). The idea that computer programming was difficult for students and teachers (Armoni, 2011; Gökçearslan & Alper, 2015) has tried to be removed via practical programs like Scratch, Alice and AppInventor that were developed for visual programming. The practicality of the environments they offer and their use of visual programming (Lye & Koh, 2014) have enabled younger students to learn the basic logic of computer programming (Kalelioğlu, 2015). The fact that only a limited number of students have been successful in learning computer programming in the past (Resnick et al., 2009) brings to mind the question, "Will the same problem occur once again?" Thus, it is required that the thoughts of students about computer programming and their success in computer programming be evaluated. Evaluating the self-efficacy levels of students is considered an important factor in terms of making an interpretation about how successful they are or will be (Aşkar & Davenport, 2009; Anastasiadou & Karakos, 2011). Even though there are various studies for measuring self-efficacy levels in different subject areas, there is a limited number of studies for measuring self-efficacy in relation to computer programming (Aşkar & Davenport, 2009). The studies that have been conducted generally comprise high school and university students (Aşkar & Davenport, 2009; Korkmaz & Altun, 2014; Mazman & Altun; 2013 Ramalingam & Wiedenbeck,1998). In this context, this study focuses on developing the "Computer Programming Self-Efficacy Scale for Secondary School Students" for determining the self-efficacy levels of younger students.

Literature review

Teaching computer programming in K-12 education

The LOGO program has been used in computer programming teaching aimed at K-12 students since the 1960s (Feurzeig & Papert, 2011, p. 487). During the 1980s, when the first personal computer was introduced, there was a demand for teaching all children how to carry out computer programming, and millions of students in thousands of schools wrote simple programs via the LOGO and Basic programs (Resnick et al., 2009). In later learning/teaching processes, computer programming teaching was conducted at various different levels. Even though various package software for teaching computer programming teaching was excluded from the teaching process despite its common usage (Kafai & Burke, 2015), it has recently been used again as a popular tool in the international arena. Computer and programming courses have also started to be taught at an early age in a number of countries (Jones, 2011). In the United States, the Computer Science Teacher Association emphasizes the importance of computational thinking and computer programming at the K-12 level, and states that these will provide skills that are needed in a number of disciplines (Seehorn et al, 2011). There have been studies conducted for teaching problem-solving skills to preschoolers aged 5-6 via wizard computer programming (Fessakis, Gouli & Mavroudi, 2013). In Turkey, while the Information Technologies and Software course used to be taught as an elective course, it has become compulsory for secondary schools as from 2012.

It has been suggested that a computer programming education is a lifelong process that not only consists of coding, but also enables students to apply the stages of problem-solving using various resources (Booth, 1992; Maheshwari, 1997). Today, students are able to construct algorithms via different computer programming instruments, in different environments and through teaching methods, and the attempt is being made to depict computer programming as a not-so-difficult process (Lewis, 2010; Resnick et al., 2009). Studies generally focus on variables regarding the motivation of students toward computer programming (Kelleher, Pausch, & Kiesler, 2007; Kelleher, & Pausch, 2007), attitudes toward computer programming (Kalelioğlu, 2015) and self-efficacy (Lee, Park & Hwang, 2013).

Self-efficacy

It is known that a number of factors are effective for success in the learning process and that self-efficacy and attitude are more important than other factors (Austin, 1987; Anastasiadou & Karakos, 2011). Self-efficacy can be defined as the perceptions of students regarding their own skills and is thought to be directly associated with their performance and effort in performing a task (Bandura, 1977). As developed within the scope of the Social Cognitive Theory, the notion of self-efficacy plays a key role in determining the emotions that affect human behaviors and performance, such as happiness, sorrow and shame (Bandura, 2001). A higher level of self-efficacy will increase the success of individuals and the level of happiness caused by that success. Individuals who trust their talents are more advanced in coping with difficult tasks (Bandura, 2001).

In measuring self-efficacy, the aim is to measure the performance capacities of individuals rather than their personal qualities (Zimmerman, 2000). It is thought that

determining the self-efficacy levels of individuals could be used as a means of increasing their success as it provides feedback about their performance (Askar & Davenport, 2009).

Measurement of self-efficacy in terms of computer programming

One of the most commonly known tools for the measurement of computer programming self-efficacy is the Computer Programming Self Efficacy Scale (CPSES), designed by Ramalingam and Wiedenbeck (1998). The scale consists of 32 items and 7-point Likert-type questions were formulated to determine the self-efficacy of students. Answers were graded from 1 ('not confident at all') to 7 ('absolutely confident'). The data collection process was carried out with 421 students in the first week of a semester at a large public university. The scale was developed for the C++ programming language. Items were collected for 4 factors in accordance with exploratory factor analysis. These factors were "(1) independence and persistence, (2) complex programming tasks, (3) self-regulation and (4) simple programming tasks" (Ramalingam and Wiedenbeck, 1998). On the full 32 item scale, reliability coefficients and empirically obtained factors as outcomes of exploratory factor analysis were determined for the scores. The reliability of test-retest was also determined. The overall alpha reliability for the scores was .98. The scores also had .50 to .84 corrected item-total correlations. The alpha reliabilities of the factors were as follows: (1) independence and persistence = .94, (2) complex programming tasks = .93, (3) self-regulation = .86, and (4) simple programming tasks = .93. Ramalingam and Wiedenbeck (1988) developed a scale for a group of novice programmers in a special programming language (C++). This scale was adapted to Turkish (Altun & Mazman, 2012). Assessment of the general programming self-efficacy levels of secondary education students is of importance in the context of the place of programming education in the K12 education program.

Aim of the study

The skill of computational thinking, which is thought to be among the necessary life skills required in the 21st century (Philips, 2009; Wing 2010), is also possibly considered to have a positive effect upon the development of other high-level thinking skills in students (Brichacek, 2014). Today, one of the methods being used in inculcating the skill of computational thinking is the teaching of computer programming. A number of countries have conducted studies in an attempt to develop the skills of computer programming in young children. Despite the positive effects and popularity of the learning computer programming skills, learning them is considered difficult by both teachers and students (Nilsen & Larsen, 2011; Caspersen & Kolling 2009; Shadiev et al., 2014). In addition to this, it has been observed that students have a low performance in computer programming courses (Aşkar & Davenport, 2009). Determining the level of self-efficacy, which is one of the indicators of performance, can be considered among the factors that would provide information about the potential computer programming performance of students. Researchers and educators have paid great attention to computer programming (Ke, 2014; Uysal & Yalın, 2012).

Additionally, Aşkar and Davenport (2009) emphasized that the perception of self-efficacy has been investigated in a number of areas, in an attempt to examine the relationship between academic success and demographic features, but that there have only been a limited number of studies regarding computer programming, which could be associated with the fact that even though there are instruments aimed at measuring the self-efficacy for different

subject areas (Compeau & Higgins, 1995; Murphy, Coover & Owen, 1989), there is a limited number of assessment instruments regarding the actual skills of computer programming. Even though the literature involves assessment instruments aimed at determining the computer programming self-efficacy levels of university students (Ramalingam & Wiedenbeck,1998), there is no assessment instrument developed specifically for secondary school students. Considering the fact that the computer programming teaching has recently become widespread at the K-12 level, it could be asserted that there is also a need for self-efficacy studies at this level. The focal point of this study comprises the development of a Computer Programming Self-Efficacy Scale for secondary school students in order to remove this deficiency in the literature.

Method

This is a scale development study. This section involves the participants, procedure and the data analysis of the scale.

Participants

This study was conducted with a study group of 233 students from the age group of 12-14 receiving education at a public secondary school in Ankara in the school year 2014-2015. According to the literature if the sample size is over then 200, it is enough for factor analyis (Büyüköztürk, 2002; Kline, 1994). Sample size -233 students- is enough for factor analysis. The scale was developed by employing statistical processes on the data that were obtained from this study group. In the group, 53.6% (n=125) were female students and 46.4% (n=108) were male students. 19.3% of students were (n=45) 5th grade (the age of 12), 59.2% (n=138) 6th grade (the age of 13) and 21.5% (n=50) 7th grade (the age of 14). Students that participated in the study were trained for programming via Scratch and SmallBasic within the scope of Information Technologies and Software lessons.

Procedure

The scale developed according to classical measurement theory. The following steps were taken in the scale development proccess (DeVellis, 2003);

- “Determine clearly what it is you want to measure
- Generate an item pool
- Determine the format for measurement
- Have the initial item pool reviewed by experts
- Consider inclusion of validation items
- Administer items to a development sample
- Evaluate the items
- Optimize scale length”

We first examined the previous scales that had been developed (Ramalingam & Wiedenbeck, 1998) and adapted (Aşkar & Davenport, 2009; Korkmaz & Altun, 2014; Altun & Mazman, 2012) to measure the computer programming self-efficacy. Then, the standarts, set by the organizations like Computer Science Teacher Association (CSTA) and International Society for Technology in Education (ISTE), were examined. Finally, an item pool was created

by writing items in accordance with the educational levels of secondary school studentsas a result of the screening that had been performed in the literature. While item pool was being created, the competencies in the National ICT Curriculum were taken into consideration. The item pool involved a total of 30 items. Items listed considering the steps used for the solution of a programming problem. We used a 5-point likert scale for expressing the level of agreement regarding the items in the scale ("*strongly agree", "agree", "undecided", "disagree", and "strongly disagree*"). In the validity study, we at first presented the content to 7 academics who had studied computer programming in the field of educational technology to check the content validity, as well as to a Turkish language linguist who is specialized in children's literature, and canvassed their opinions via Expert Opinion Form. Experts were asked to mark if it is appropriate or not for every item in the Expert Opinion Form. According to the opinions and criticisms received, we made the required corrections, additions and deletions from the scale items, formed a scale of a total of 33 items and conducted the validity and reliability studies on the basis of these items. All participants participated in the study on a voluntary basis.

Data analysis

The following analyses were performed in an attempt to prove the validity and reliability of the data obtained from 233 secondary school students:

- Kaiser-Meyer Olkin (KMO) coefficient and Barlett's Sphericity test for determining the fit of the data for the factor (principal components) analysis.
- Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA) for proving the construct validity.
- Parallel analysis for deciding on the sub-factor number of the scale.
- Cronbach's Alpha and Equivalent Halves method reliabilities for proving the reliability.
- Item test correlations for proving the item validity.

Results

In the study, the statistical processes, exploratory factor analysis and confirmatory factor analysis were performed sequentially.

Findings regarding the fit for the factor (principal components analysis)

The Kaiser-Meyer-Olkin (KMO) coefficient is a statistical method used in determining whether or not the sample is suitability for conducting a factor analysis (Kan & Akbaş, 2005). For this purpose, we determined the KMO value as 0.949. The minimum KMO value of 0.6 is suggested for conducting a factor analysis upon data (Özel, Timur, Timur & Bilen, 2013; Pallant, 2010). Secondly, we checked the Bartlett Sphericity test ($\chi 2$ = 3532.449, p.=0.000). The fact that the Chi-square test acquired as a result of the Barlett Sphericity test was significant indicates that the data come from a multivariate normal distribution (Kan & Akbaş, 2005). According to these results, it was observed that the exploratory factor analysis could

be performed via the acquired data. In Table 1, range, min, max, mean, standard deviation, skewness, kurtosis values of the items are given.

Table 1

Descriptive statistics about Scale Items

Item	N	Range	Min	Max	Mean	S.D.	Skewness	Kurtosis
I1	233	4.00	1.00	5.00	3.7597	1.10356	-.754	.030
I2	233	4.00	1.00	5.00	3.5451	1.07835	-.554	-.232
I3	233	4.00	1.00	5.00	3.8584	1.08740	-.953	.385
I4	233	4.00	1.00	5.00	3.9657	1.10976	-1.115	.728
I5	233	4.00	1.00	5.00	4.1459	1.10453	-1.454	1.558
I6	233	4.00	1.00	5.00	3.8369	1.09427	-.906	.260
I7	233	4.00	1.00	5.00	3.8155	1.06889	-.822	.133
I8	233	4.00	1.00	5.00	3.5794	1.13103	-.614	-.325
I9	233	4.00	1.00	5.00	3.8197	1.09156	-.840	.104
I10	233	4.00	1.00	5,00	3.8412	1.08104	-.898	.336
I11	233	4.00	1.00	5.00	3.3519	1.27499	-.307	-.963
I12	233	4.00	1.00	5.00	3.7682	1.26880	-.833	-.298
I13	233	4.00	1.00	5.00	3.4893	1.16008	-.483	-.475
I14	233	4.00	1.00	5.00	3.7124	1.07024	-.746	.015
I15	233	4.00	1.00	5.00	3.6137	1.14703	-.705	-.200
I16	233	4.00	1.00	5.00	3.6438	1.13607	-.529	-.493
I17	233	4.00	1.00	5.00	3.6266	1.14943	-.674	-.327
I18	233	4.00	1.00	5.00	3.5236	1.14493	-.293	-.655
I19	233	4.00	1.00	5.00	3.8026	1.24381	-.799	-.348
I20	233	4.00	1.00	5.00	3.7983	1.13624	-.841	.035
I21	233	4.00	1.00	5.00	3.7296	1.09852	-.667	-.120
I22	233	4.00	1.00	5.00	3.8498	1.16660	-.854	-.071
I23	233	4.00	1.00	5.00	3.5794	1.05614	-.421	-.245
I24	233	4.00	1.00	5.00	3.6609	1.16747	-.738	-.108
I25	233	4.00	1.00	5.00	3.8712	1.09493	-.775	-.043
I26	233	4.00	1.00	5.00	3.5622	1.06137	-.448	-.205
I27	233	4.00	1.00	5.00	4.0215	1.16503	-1.048	.185
I28	233	4.00	1.00	5.00	3.8026	1.21931	-.839	-.236
I29	233	4.00	1.00	5.00	3.5322	1.16709	-.480	-.445
I30	233	4.00	1.00	5.00	3.7210	1.15009	-.741	-.159
I31	233	4.00	1.00	5.00	3.8584	1.13397	-.881	.084
Average	233	4.00	1.00	5.00	3.7318	.72310	-1.058	1.641

Findings regarding the exploratory factor analysis

As a result of the exploratory factor analysis, we initially excluded two items because they placed a similar load on more than one factor within the scale of 33 items. The final form of the scale involves 31 items.

It could be suggested that factors equal to the number of components with Eigen values larger than 1 should be included (Çokluk, Şekercioğlu & Büyüköztürk, 2010). Examining the Total Variance Distribution Being Explained in Table 1, we could observe six factors with Eigen values larger than 1. However, according to the parallel analysis method (Pallant, 2010), alternatively used in determining the number of factors (especially for the scales developed for social sciences studies), the number of factors was determined as 1. “In this analysis, a program is used that is called Monte Carlo PCA for Parallel Analysis. In this program you are asked for three pieces of information: the number of variables you are analysing (number of items); the number of participants in your sample; andthe number of replications (specify 100).After that, this generates 100 sets of random data of same size as real data file. It will calculate the average eigenvalues for these 100 randomly generated samples and print these out for you. After that you compare the eigenvalues obtained from SPSS and Monte Carlo PCA for Paralel Analysis. If your value is larger than the criterion value from parallel analysis, you retain this factor; if it is less, you reject it” (Pallant, 2010, p.194). While Table 2 shows total variance distributions, Table 3 shows the results of the parallel analysis, Table 4 shows the findings regarding the item factor loads and test correlations.

Table 2

Total Variance Distributions Being Explained

Component	Initial Eigenvalues			Extraction sums of squared loadings		
	Total	% of variance	Cumulative %	Total	% of variance	Cumulative %
1	12.756	41.150	41.150	12.756	41.150	41.150
2	1.349	4.351	45.501			
3	1.240	4.001	49.501			
4	1.134	3.658	53.159			
5	1.044	3.368	56.527			
6	1.017	3.280	59.807			

Examining Table 2, the scale being evaluated under one factor shows variance at the rate of 41.15%, which is acceptable for one factor structure (> 30%) (Büyüköztürk, 2010; Çokluk, Şekercioğlu & Büyüköztürk, 2010; Tabachnick & Fidell, 1996). There is no exact threshold value of the explained total variance in the EFA test for all practices. In social sciences although 60 % explained total variance is frequently encountered, this value can be lower (Hair, Anderson, Tatham & Black, 1998). Rotating procedure simplifies the factor structure (Abdi, 2003). Rotated structure attempts to have each variable load on as few factors as possible (Yong & Pearce, 2013). The scale is formad by a single factor that’s way rotation is not performed.

Table 3

Comparison of Eigenvalues from PCA and Criterion Values from Parallel Analysis

Component number	Actual eigenvalue from PCA	Criterion value from parallel analysis	Desicion
1	12.756	1.7498	Accept
2	1.349	1.6380	Reject
3	1.240	1.5644	Reject
4	1.134	1.4979	Reject
5	1.044	1.4388	Reject
6	1.017	1.3825	Reject

According to the Table 3, just one dimention is accepted. The test results proves that the scale is one dimensional

Table 4

Factor Load Distribution Values and Item test correlations of the Programming Self-Efficacy Scale for Secondary School Students

Items***	Factor load	Corelation	Total scale correlation	Cronbach's Alpha if Item Deleted
I24: I can enable the program to produce accurate results.	0.718	Pearson Correlation Significance (2-tailed)	0.712** 0.000	0.949
I6: I can solve the problem via different solutions.	0.709	Pearson Correlation Significance (2-tailed)	0.702** 0.000	0.949
I16: I know how to use the programming variables.	0.708	Pearson Correlation Significance (2-tailed)	0.707** 0.000	0.949
I22: I can operate the program I have developed.	0.707	Pearson Correlation Significance (2-tailed)	0.702** 0.000	0.949
I27: I can record the program I have developed.	0.703	Pearson Correlation Significance (2-tailed)	0.699** 0.000	0.949
I31: I can explain my idea of software project step by step.	0.695	Pearson Correlation Significance (2-tailed)	0.691** 0.000	0.949
I30: Among the multiple software projects, I select the one that is the fittest for the criterion.	0.693	Pearson Correlation Significance (2-tailed)	0.691** 0.000	0.949
I5: I select the fittest knowledge for solving the programming problem.	0.693	Pearson Correlation Significance (2-tailed)	0.668** 0.000	0.949
I4: I investigate the knowledge that is required for solving the programming problem.	0.691	Pearson Correlation Significance (2-tailed)	0.610** 0.000	0.949

Items***	Factor load	Corelation	Total scale correlation	Cronbach's Alpha if Item Deleted
I10: Among various steps of solution, I select the fittest one for the solution to the programming problem.	0.688	Pearson Correlation Significance (2-tailed)	0.683** 0.000	0.949
I7: I can determine the fittest solution to a problem.	0.688	Pearson Correlation Significance (2-tailed)	0.684** 0.000	0.949
I25: I can make changes on the program.	0.674	Pearson Correlation Significance (2-tailed)	0.667** 0.000	0.949
I15: I can make preparations (like determining the variables and processes) required for solving the programming problem.	0.670	Pearson Correlation Significance (2-tailed)	0.668** 0.000	0.949
I3: I can make an interpretation regarding whether or not a programming problem could be solved.	0.656	Pearson Correlation Significance (2-tailed)	0.608** 0.000	0.949
I8: I can suggest different solutions in order to solve the programming problems.	0.650	Pearson Correlation Significance (2-tailed)	0.519** 0.000	0.949
I26: I can correct the mistakes about the coding in the program.	0646	Pearson Correlation Significance (2-tailed)	0.644** 0.000	0.949
I9: I determine the solution to the programming problem step by step.	0639	Pearson Correlation Significance (2-tailed)	0.634** 0.000	0.950
I20: I know the stages of programming.	0.639	Pearson Correlation Significance (2-tailed)	0.637** 0.000	0.950
I29: I can explain the process of developing a software project.	0.637	Pearson Correlation Significance (2-tailed)	0.644** 0.000	0.950
I17: When necessary, I can change the order of the processes designed for solving a programming problem.	0.636	Pearson Correlation Significance (2-tailed)	0.642** 0.000	0.950
I28: I can share my program with other people via the internet.	0.628	Pearson Correlation Significance (2-tailed)	0.632** 0.000	0.950
I23: I can enable the perfect functioning of the program.	0.612	Pearson Correlation Significance (2-tailed)	0.613** 0.000	0.950
I14: I can discuss the different steps being developed for solving the programming problem.	0.609	Pearson Correlation Significance (2-tailed)	0.610** 0.000	0.949

Items***	Factor load	Corelation	Total scale correlation	Cronbach's Alpha if Item Deleted
I13: I can correct a programming problem whose solution steps are given wrong.	0.605	Pearson Correlation Significance (2-tailed)	0.608** 0.000	0.950
I21: I know where to write the program codes.	0.589	Pearson Correlation Significance (2-tailed)	0.589** 0.000	0.950
I12: I share the steps of solution to the programming problem with my friends.	0.584	Pearson Correlation Significance (2-tailed)	0.593** 0.000	0.950
I2: I can solve complex programming problems by separating them into smaller sub-problems.	0.582	Pearson Correlation Significance (2-tailed)	0.586** 0.000	0.950
I1: I can understand whether a problem is a programming problem or not.	0.546	Pearson Correlation Significance (2-tailed)	0.549** 0.000	0.950
I19: I know what the operators +, -, *, /, >,<, = mean in a programming.	0.512	Pearson Correlation Significance (2-tailed)	0.523** 0.000	0.951
I18: I can use the cycle instead of repeating instructions.	0.508	Pearson Correlation Significance (2-tailed)	0.519** 0.000	0.951
I11: I can show the steps of solution by drawing figures on paper.	0.473	Pearson Correlation Significance (2-tailed)	0.492** 0.000	0.951

*The table does not involve the load values of items as .40 and lower (Büyüköztürk, 2002).

**Correlation is significant at the level of 0.01 (2-tailed).

***The scale developed in Turkish. All items are translated into English for this article.

Table 4 shows the factor load distribution values of the scale. The factor loads of the scale, which involves a single factor of 31 items, obtained values varying between 0.473 and 0.718. it can be observed that all the items in the scale have a moderate and high relationship with the total scale score having a significance level of 0.01 ($p<0.01$). The item test correlations of the scale have values between 0.492 and 0.712. The correlation values for the item validity and homogeneity of the scale prove that the scale items are valid and measure the same structure. Examining the item test correlation values, it is observed that the scale items have a sufficient validity level.

Findings regarding the confirmatory factor analysis

The fit of model regarding the factor structure presented as a result of the Exploratory Factor Analysis (EFA) was tested via Confirmatory Factor Analysis (CFA). The fit of the acquired model was tested via the cohesion criterion of X^2/df, RMSEA (Root Mean Square Error

Approximation), NFI (Normed Fit Index), NNFI (Non-Normed Fit Index), RMR (Root Mean Square Residual), CFI (Comparative Fit Index), IFI (Incremental Fit Index), GFI (Goodness of Fit Index) and AGFI (Adjusted Goodness of Fit Index). As a result of the analysis, we determined the fit of model as X^2/df = 1.84; RMSEA value = 0.06; NFI value = 0.95; NNFI value = 0.98; RMR value = 0.068; CFI value = 0.98; IFI value = 0.98; GFI value = 0.82 and AGFI value = 0.79. Considering the data acquired, it was observed that some of the fit values of the model were not acceptable. Modifications suggested as a result of this analysis were implemented. As a result of the modifications, we determined X^2/df = 1.47; RMSEA value = 0.045; NFI value = 0.96; NNFI value = 0.99; RMR value = 0.061; CFI value = 0.99; IFI value = 0.99; GFI value = 0.85 and AGFI value = 0.83.

The majority of goodness of fit indexes have a value between 0 and 1. While the value 0 signifies that there is no fit between the data and the model, the value 1 signifies that there is a perfect fit. If the value of an index is larger than 0.9 and is almost 1, it can be asserted that the data is an almost perfect fit (Çerezci, 2010). Şimşek (2007) suggests that if the χ^2/df value is 5 or lower and the RMSEA value is 0.08 or lower, there is a good fit. Byrne (1998), on the other hand, suggests that a good fit requires the RMR and SRMR values to be 0.1 or lower. Similarly, a good fit requires the IFI, CFI, NFI and NNFI to be greater than 0.9. In addition to this, if the AGFI is 0.8 or greater and the GFI is 0.85 or greater, this signifies an acceptable fit (Çokluk, Şekercioğlu & Büyüköztürk, 2010). Considering the goodness of fit indexes acquired within the scope of this study, it can be observed that the scale has a statistically acceptable goodnes of fit. While Table 5 shows the fit indexes of the scale involving a single factor and 31 items before and after modification, Figure 1 shows the Structural Equation Model and the Standard Values after modification.

Table 5.

Fit Values of the Programming Self-Efficacy Scale for Secondary School Students

Fit index	Before the modification	After the modification	Acceptable value
Chi-Square (X^2)	796.96	618.32	
Degree of Freedom	434	422	
Chi-Square/df	1.84	1.47	<=5
RMSEA	0.06	0.045	<=0.08
NFI	0.95	0.96	>0.9
NNFI	0.98	0.99	>0.9
RMR	0.068	0.061	<=0.1
CFI	0.98	0.99	>0.9
IFI	0.98	0.99	>0.9
GFI	0.82	0.85	>=0.85
AGFI	0.79	0.83	>=0.8

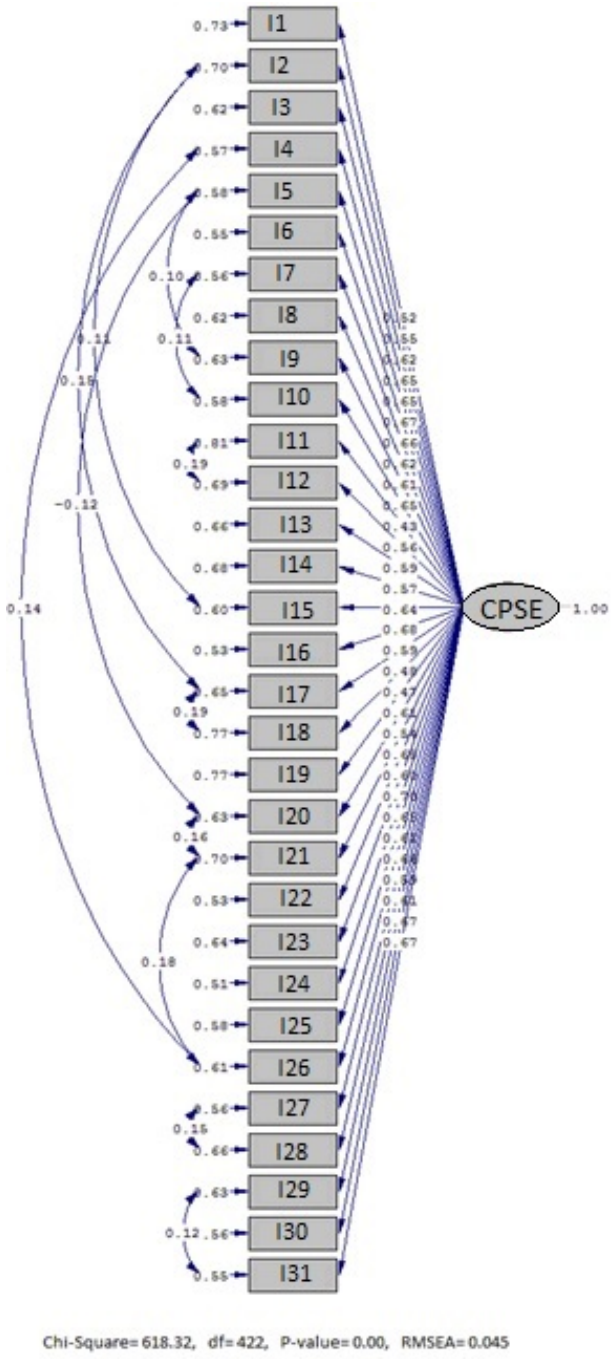

Figure 1. Structural equation model and the standard values after modification

Findings regarding the reliability coefficient using the method of reliability analysis and equivalent halves (two halves)

As a result of the Cronbach alpha reliability analysis, the reliability coefficient of the scale was determined to be as high as 0.95 (Özdamar, 1999). The scale items were separated into two parts consisting of sole and double items and the reliability analysis was conducted via the method of equivalent halves (completing half of the test). The Pearson Correlation table between the first and the second halves is as follows (Table 6).

Table 6

Correlation Values Being Acquired Via the Method of Completing Half of the Test Regarding the Programming Self-Efficacy Scale for Secondary School Students

		First half	Second half
First half	Pearson Correlation	1	0.935*
	Significance (2-tailed)		0.000
	N	233	233

*. Correlation is significant at the level of 0,01 (2-tailed).

The reliability coefficient of the entire test could be determined via the formula $rentire = \frac{2*rpearson}{1+rpearson}$ using the Spearman-Brown method (Ellez, 2009). From this point of view, we determined the reliability coefficient of the entire test via the Spearman Brown method as follows: $rentire = \frac{2*0.935}{1+0.935}$ = 0.966. The relationship between the first and the second half of the test was determined as statistically and positively significant at the level of p<0.01 (see Table 6).

Discussion

Computer programming for children has recently been included in numerous curricula, with various courses employing various teaching practices, and it has an increasing popularity worldwide. However, it has been asserted that the popularity of educational computer programming for children has only been reflected in a limited way in research about this teaching (Fessakis et al., 2013). In particular, there is only a limited number of studies for determining the self-efficacy level of children with regard to computer programming (Aşkar & Davenport, 2009).

In this study, an instrument was developed to measure the self-efficacy levels of secondary school students regarding computer programming and the psychometric features of the scale were examined. The steps of scale development were followed. The item pool of the scale, consisting of 30 items, was evaluated in accordance with expert opinion. Items were excluded and added according to the feedback received from the experts. In the pilot application, 33 items were presented to the secondary school students.

As a result of the study, a unidimensional scale of 31 items and a 5-point likert scale was presented. In the unidimension, the factor loads varied between 0.47 and 0.71. The variance rate for this scale structure was 41.15%. It can be asserted that the variance explained by the scale structure explains why it is able to measure sufficiently.

Examining the fit indexes of the scale structure, we determined the X2 / df = 1.84; RMSEA value = 0.06; NFI value = 0.95; NNFI value = 0.98; RMR value = 0.068; CFI value = 0.98; IFI value = 0.98; GFI value = 0.82 and AGFI value = 0.79. As some values were not acceptable for the goodness of fit of the model, we implemented the modifications suggested and determined X^2 / df = 1.47; RMSEA value = 0.045; NFI value = 0.96; NNFI value = 0.99; RMR value = 0.061; CFI value = 0.99; IFI value = 0.99; GFI value = 0.85 and AGFI value = 0.83. All the goodness of fit indexes showed an acceptable fit (Byrne, 1998). From this perspective, the scale structure was observed to have an acceptable fit.

Consistency-related evidence was obtained for the reliability of the computer programming self-efficacy scale for children. The Cronbach alpha for internal consistency, involving all 31 items, was determined as 0.95. From various methods aimed at determining the internal consistency of the scale, we used the equivalent halves method. As a result of the reliability analysis that was performed via the equivalent halves method, we obtained a value of 0.966. The fact that these values are acceptable reliability values shows that the scale had a sufficient internal consistency level.

Teachers and researchers could use the CPSES as a single factor structure to measure the computer programming self-efficacy levels of students in teaching with programs like Scratch, Logo, Alice that have commonly been used, especially in recent years, to educate children in computer programming.

Conclusion

Today, in parallel with developments in information technology, computer programming has become an important area, attracting large financial investments worldwide. Computers, mobile computers and smart phones are equipped with constantly evolving software that meets different needs with each passing day. Furthermore, it has been suggested that computational thinking should be considedered among the basic skills required in the 21st century (Philips, 2009; Wing 2010). Reading, writing and arithmetic have always been among the basic skills, but today individuals of the 21st century also need to have the ability to think like computer scientists in order to solve ever more complex problems and carry out required tasks (Wing, 2006). Computer science has an interdisciplinary relationship with other disciplines (Barr, & Stephenson, 2011). Yet teaching and learning computer programming are still considered difficult for both students and educators (Black, 2006; Shadiev et al., 2014). In order to increase the degree of computer programming self-efficacy, its initial level must first be determined. There is, however, an extremely limited number of assessment instruments aimed at measuring computer programming self-efficacy in children. It is possible to assert that the assessment instrument developed within the scope of this study can measure computer programming self-efficacy in a valid and reliable way and this instrument is thus useful in terms of responding to and filling the lack of a relevant assessment instrument.

The results of this study could be generalized, although based on some limitatons. The study group comprises only secondary school students. In addition to this, the confirmatory factor analysis was conducted via the exploratory factor analysis data. “Both exploratory and confirmatory techniques are useful tools for analyzing the complex data sets” (Plucker, 2003). “If a good fit is questionable when the factor structure is confirmatively tested on the same data, we cannot expect that a test of the factor structure in a confirmative follow-up study, that is, on different data, will lead to a good fit” (Van Prooijen & Van Der Kloot, 2001). Another limitation of the study is that it did not examine the external criterion validity within the scope of validity studies. Apart from these limitations, it was observed that the assessment instrument was able to adequately measure the structure of the scale.

It is suggested that there be further research into whether the assessment instrument developed here has a similar validity in assessing high school and university students, as well as within different languages and cultures. It is also suggested that studies be conducted

regarding any personal and demographic features of students which may affect their levels of computer programming self-efficacy, as well as the educational methods and techniques that may also have an effect. It should be possible to determine the obstacles that negatively affect computer programming teaching and to conduct comprehensive studies that could increase the success in this subject. It is suggested that studies be conducted that would model the relationships between self-efficacy in computer programming and other problem-solving skills, as well as other critical and high-level thinking and computational thinking skills. Moreover, such studies could determine and support the contribution of computer programming skills to the overall cognitive development of students. It is also suggested that the effect of computer programming on the teaching and learning of maths and other subjects be investigated.

References

Abdi, H. (2003). *Factor rotations in factor analyses. Encyclopedia for Research Methods for the Social Sciences.* Sage: Thousand Oaks, CA, 792-795.

Altun, A., & Mazman, S. G. (2012). Programlamaya ilişkin öz yeterlilik algısı ölçeğinin Türkçe formumun geçerlilik ve güvenirlik çalışması. *Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi, 3*(2), 297-308.

Anastasiadou, S.D., & Karakos, A.S. (2011). The beliefs of electrical and computer engineering students regarding computer programming. *The International Journal of Technology, Knowledge and Society*, *7*(1), 37-51.

Armoni, M. (2011). The nature of CS in K-12 curricula: the roots of confusion. ACM Inroads, 2(4), 19-20. doi:10.1145/2038876.2038883

Askar, P., & Davenport, D. (2009). .An investigation of factors related to self-efficacy for java Programming among engineering students. *The Turkish Online Journal of Educational Technology TOJET, 8*(1): 26-32.

Austin, H.S. (1987). Predictors of pascal programming achievement for community college students. *Proceedings of the eighteenth SIGCSE technical symposium on Computer science education*, Missouri, United States, 161-164. doi: 10.1145/31726.31752

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review, 84*, 191-215, http://dx.doi.org/10.1037/0033-295X.84.2.191

Bandura, A. (2001). Social cognitive theory: An agentic perspective. *Annual review of psychology*, *52*(1), 1-26. doi:10.1146/annurev.psych.52.1.1

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. *ACM Inroads, 2*(1), 48-54. doi:10.1145/1929887.1929905

Black, T.R. (2006). Helping novice programming students succeed. *Journal of Computing Sciences in Colleges, 22*(2), 109–114.

Booth, S. (1992). *Learning to program: A phenomenographic perspective*. University of Gothenburg Publication, http://hdl.handle.net/2077/16224

Brichacek, A. (2014). Computational thinking boosts students' higher-order skills. Retrieved May 21, 2015 from https://www.iste.org/explore/articleDetail?articleid=232&category=Featured-videos&article=Computational%20thinking%20boosts%20students%E2%80%99%20higher-order%20skills.

Büyüköztürk, Ş. (2002). Faktör analizi: Temel kavramlar ve ölçek geliştirmede kullanımı. *Kuram ve uygulamada eğitim yönetimi, 32*(32), 470-483.

Büyüköztürk, Ş. (2010). *Sosyal bilimler için veri analizi el kitabı* [Handbook of data analysis for the social sciences], Ankara: Pegem Akademi.

Byrne, B. M. (1998). *Structural equation modeling with lisrel, prelis and simplis: basic concepts, applications, and programmings*. London: Lawrence Erlbaum Assocatiates, Publishers.

Caspersen, M. E., & Kolling M. (2009). STREAM: A first programming process. *ACT Transaction on Computing Education, 9*, 1-29. doi:10.1145/1513593.1513597

Compeau, D. R., & Higgins, C. A. (1995). Computer self-efficacy: Development of a measure and initial test. *MIS quarterly*, 189-211, http://www.jstor.org/stable/249688

Çerezci, E.T. (2010). *Yapısal eşitlik modelleri ve kullanılan uyum iyiliği indekslerinin karşılaştırılması.* (Unpublished Doctoral Dissertation). Gazi Üniversitesi Fen Bilimleri Enstitüsü: Ankara.

Çokluk, Ö., Şekercioğlu, G., &Büyüköztürk, Ş. (2010). *Sosyal bilimler için çok değişkenli istatistik: SPSS ve LISREL uygulamaları*. Ankara: Pegema Yayıncılık.

DeVellis, R. F. (2012). *Scale development: Theory and applications* (Vol. 26). London: Sage publications.

Ellez, A. M. (2011). Ölçme araçlarında bulunması gereken özellikler. *Bilimsel araştırma yöntemleri. (In Second Edition)*, 165-190. Ankara: Anı Yayıncılık.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87-97. doi: 10.1016/j.compedu.2012.11.016

Fessakis, G., & Serafeim, K. (2009). Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education. In *ACM SIGCSE Bulletin* (Vol. 41, No. 3, pp. 258-262). ACM. Doi: 10.1145/1595496.1562957

Feurzeig, W., & Papert, S. A. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments, 19*(5), 487-501.doi: 10.1080/10494820903520040

Gökçearslan, Ş., & Alper, A. (2015). The effect of locus of control on learners' sense of community and academic success in the context of online learning communities. *The Internet and Higher Education*, *27*, 64-73. Doi: 10.1016/j.iheduc.2015.06.003

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43. doi: 10.3102/0013189X12463051

Hair, J. F., Anderson, R. E., Tatham, R. L., & Black, W. C. (1998). *Multivariate data analysis*. PrenticeHall International, Upper Saddle River, New Jersey.

ISTE. (2007). ISTE standards students. International Society for Technology in Education: Retrieved, August, 2015 from https://www.iste.org/docs/pdfs/20-14_ISTE_Standards-S_PDF.pdf

Jones, S. P. (2011). Computing at School International comparisons. Retrieved Ağustos 5, 2015 from http://www.computingatschool.org.uk/index.php?id=documents adresinden.

Kafai, Y., & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan, 95*(1), 61–65.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior*, *52*, 200-210. doi:10.1016/j.chb.2015.05.047

Kan, A., & Akbaş. A. (2005). Lise öğrencilerinin kimya dersine yönelik tutum ölçeği geliştirme çalışması. *Mersin Üniversitesi Eğitim Fakültesi Dergisi,* 1 (2), 227-237.

Kay, R. H., & Knaack, L. (2005). A case for ubiquitous, integrated computing in teacher education. *Technology, Pedagogy and Education*, *14*(3), 391-412. doi:10.1080/14759390500200213

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education, 73*, 26–39. doi:10.1016/j.compedu.2013.12.010

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM*, *50*(7), 58-64. Doi: 10.1145/1272516.1272540

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 1455-1464). ACM. doi: 10.1145/1240624.1240844

Kline, P. (1994). *An Easy Guide To Factor Analysis*. New York: Routledge

Korkmaz, Ö., & Altun, H. (2014). Adapting computer programming self-efficacy scale and engineering students' self-efficacy perceptions. *Participatory Educational Research (PER), 1*(1), *20-31,* http://dx.doi.org/10.17275/per.14.02.1.1

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the Computational Thinking Scales (CTS). *Computers in Human Behavior,* http://dx.doi.org/10.1016/j.chb.2017.01.005

Lee, J., Park, J. G., & Hwang, Y. (2013). A study on general and specific programming self-efficacy with antecedents from the social cognitive theory. *Journal of Next Generation Information Technology*, *4*(8), 423-432.

Lewis, C. M. (2010). How programming environment shapes perception, learning and goals: logo vs. scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 346-350). ACM. doi: 10.1145/1734263.1734383

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, *41*, 51-61. Doi: 10.1016/j.chb.2014.09.012

Maheshwari, P. (1997, July). Teaching programming paradigms and languages for qualitative learning. In *Proceedings of the 2nd Australasian conference on Computer science education* (pp. 32-39). ACM. doi:10.1145/299359.299365

Mazman, S. G., & Altun, A. (2013). Programlama-I dersinin böte bölümü öğrencilerinin programlamaya ilişkin öz yeterlilik algıları üzerine etkisi. *Journal of Instructional Technologies & Teacher Education*, *2*(3), 24-29.

Murphy, C. A., Coover, D., & Owen, S. V. (1989). Development and validation of the computer self-efficacy scale. *Educational and Psychological measurement*, *49*(4), 893-899. doi: 10.1177/001316448904900412

Nilsen H., & Larsen A. (2011). Using the personalized system of instruction in an introductory programming course. *NOKOBIT*, 27-38. November 21-23.

Özdamar, K. (1999). *Paket Programlar İle İstatistiksel Veri Analizi 1*. Eskişehir: Kaan Kitabevi.

Özel, M., Timur, B., Timur, S. & Bilen, K. (2013). Öğretim elemanlarının pedagojik alan bilgilerini değerlendirme anketinin Türkçeye uyarlanması çalışması*. Ahi Evran Üniversitesi Kırşehir Eğitim Fakültesi Dergisi (KEFAD),* 14 (1), 407-428.

Pallant, J. (2010). A step by step guide to data analysis using the SPSS program. Australia: Allen and Unwin Books.

Phillips, P. (2009). *Computational thinking a problem solving tool for every classroom.* Computer Science Teacher Association. Retrieved August 2015 from http://csta.acm.org/Resources/sub/ResourceFiles/CompThinking.pdf.

Plucker, J. A. (2003). Exploratory and confirmatory factor analysis in gifted education: Examples with self-concept data. *Journal for the Education of the Gifted, 27*(1), 20-35.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 367-381. Doi: 10.2190/C670-Y3C8-LTJ1-CT3P

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67. doi: 10.1145/1592761.1592779

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., ... & Verno, A. (2011). CSTA K--12 Computer Science Standards: Revised 2011. ACM.

Shadiev, R., Hwang, W. Y., Yeh, S. C., Yang, S. J., Wang, J. L., Han, L., & Hsu, G. L. (2014). Effects of unidirectional vs. reciprocal teaching strategies on web-based computer programming learning. *Journal of Educational Computing Research*, *50*(1), 67-95. doi:10.2190/EC.50.1.d

Şimşek, Ö. F. (2007). *Yapısal eşitlik modellemesine giriş, temel ilkeler ve LISREL uygulamaları*. Ankara: Ekinoks Yayıncılık.

Tabachnick, B. G. ve Fidell, L.v S. (1996). *Using multivariate statistics* (3. Ed.). New York: Harper Collins College Publishers.

Uysal, M. P., & Yalın, H. İ. (2012). Öğretim etkinlikleri kuramına göre tasarlanan öğretim yazılımının akademik başarıya etkisi*. International Journal of Human Sciences, 9*(1), 185–204.

Van Prooijen, J. W., & Van Der Kloot, W. A. (2001). Confirmatory analysis of exploratively obtained factor structures. *Educational and Psychological Measurement, 61*(5), 777-792.

Yong, A. G., & Pearce, S. (2013). A beginner's guide to factor analysis: Focusing on exploratory factor analysis. *Tutorials in Quantitative Methods for Psychology, 9*(2), 79-94.

Weinberg, A. E. (2013). *Computational thinking: An investigation of the existing scholarship and research. (Unpublished Doctoral Thesis)*, Colorado State University, School of Education, Colorado.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society*, 3717-3725. doi: 10.1098/rsta.2008.0118

Wing, J. M. (2010). *Computational thinking: What and Why?* Center for Computational Thinking Carnegie Mellon: Retrieved, May 2014 Retreived from https://www.cs.cmu.edu/~CompThink/papers/TheLinkWing.pdf

Zimmerman, B. J. (2000). Self-efficacy: An essential motive to learn. *Contemporary Educational Psychology*, *25*(1), 82-91. doi:10.1006/ceps.1999.1016